Design by Continuous Collaboration Between Manual and Automatic Optimization

K.E. Shahroudi

Software Engineering (SEN)

SEN-R9701 February 28, 1997

Report SEN-R9701
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Design by Continuous Collaboration Between Manual and Automatic Optimization

Kamran Eftekhari Shahroudi
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Email: kash@cwi.nl

ABSTRACT

Numerical optimization is traditionally viewed as a “machine centric” activity. This view dominates the majority of numerical optimization packages today, where user interaction is normally limited to the problem definition phase or visualization of the results with little or no interaction at design or run time.

Surprisingly we are surrounded by many examples of successful engineering systems which allow human interaction at run time, e.g. automobiles, aircraft etc. In fact, Integrated Human Machine Systems (IHMS) and dedicated engineering design groups have already shown that the distribution of the intelligent function between the human and artificial agents at design time leads to a more effective utilization of their complementary capabilities.

This paper discusses the implementation of a generic semi-automatic optimization concept in which the human designer continuously collaborates with a numerical agent to navigate the design space and modify it when necessary. The concept allows human interaction at various levels of automation.

The potential of this approach is shown by way of three human-in-the-loop optimization examples:

· conceptual design optimization of subsonic aircraft;

· optimization of trajectory of a Mars rover vehicle;

· configuration optimization of a multistage rocket.

This technique is located in a multidisciplinary area formed by the cross section of Visualization, Numerical Optimization and Computational Steering.

1991 Mathematics Subject Classification: 65K10, 90C29, 90C31, 93B51.
1991 Computing Reviews Classification System: B.5.2, C.4., D.4.7, F.1.2., J.6.
Keywords and Phrases: Collaborative Manual & Automatic Agents, Multidisciplinary Design Optimization, Human-in-the-Loop, Computational Steering, Semi-Automatic or Interactive Optimization, Natural Design Cycle.
Note: To appear in the Journal of Computer Modelling and Simulation in Engineering (CMSE).
Work carried out under project SEN 1.3 Interactive Visualization Environments.

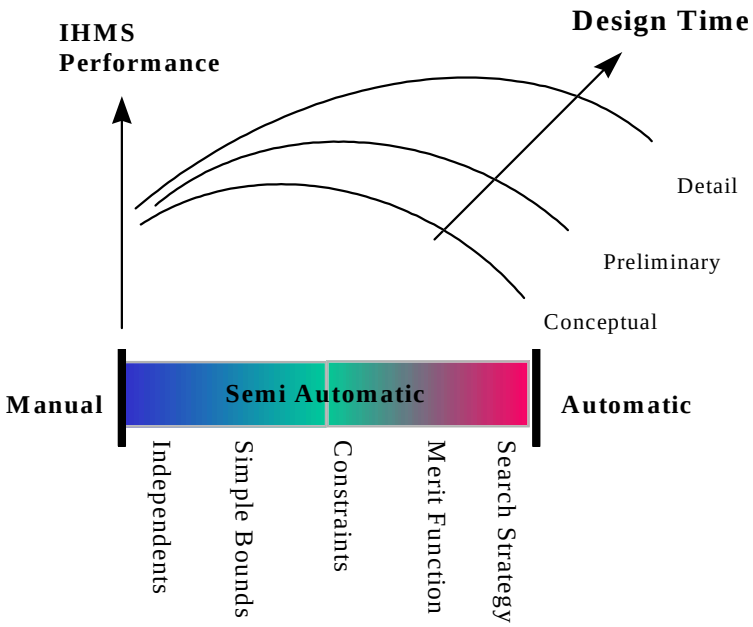

Figure 1: The automation Spectrum.

1. INTRODUCTION

The inability to perform intensive numerical computation, limited memory capacity and an upper bound on the complexity that the human designer can process have resulted in dominant "machine centered" view of design optimization as pointed out by *Boy et al.(1990).*

In principle one can draw an automation or interaction spectrum for optimization as shown in *Fig. (1)* which is an extended version of that given by *Boy (1986)*. A given position in this spectrum represents the level of automation, which is defined in terms of the number of design variables under explicit and implicit control of the human designer:

$$A = N_{\text{implicit}} \Big/ (N_{\text{implicit}} + N_{\text{explicit}})$$

Fig. (1) suggests that the position in the automation spectrum or the value of *A* depends on whether the user can interact with the independent (or design) variables, constraints, merit function and search strategy inside the optimization loop.
Generally the total number of variables increases as design continues and becomes more detailed. The figure is qualitative because the shape of the curves depend on many…

factors such as the quality of the human designer and computational resources. Also the design doesn't always follow the conceptual, preliminary and detail phases in that order.

An ideal set up would allow the full range including the fully manual and fully automatic bounds. Most implementations operate at the right bound or allow the optimization to switch back and forth between the two bounds. The focus of this paper is on the gray area in between which is rarely applied in design optimization. The figure qualitatively shows that the performance of the IHMS design system varies between the different phases of design. The optimum level of automation depends on several factors such as such as the current focus, designers knowledge, quality of design criteria and computational resources, which are all variables. As design proceeds, the number of variables increases and the quality of design criteria improves which in turn shift the optimum to the right.

It is generally agreed that design is an iterative process once a tentative design object is available. Total exclusion of the human from the iterative loop has the direct consequence that the design problem be stated in a mathematical expression, i.e. requires explicit specification of the design criteria such as design space, constraints and merit function. This happens to be the "key" problem in optimization as in real applications, it is extremely difficult (if not impossible) to define function(s) that measure the "true merit" of a design object. For example, even the most prominent aircraft designers would not dare to claim that a particular function measures the "true merit" of a class of aircraft. They may have a set of functions that can aid the design process, e.g. direct operating costs, payload fraction, reliability etc. However the trade-off between them that leads to the true optimum aircraft is always debatable and can change with time.

Imperfection in design criteria implies that the results of numerical optimization can also not be perfect requiring the human designer to improve these criteria by using intuition and experience. In most current setups, the human designer has to wait at the right bound in *Fig. (1)* until the numerical results are available (perhaps hours or days later) before being allowed to switch back to the fully manual mode to modify the deign object and criteria. *Boy et al (1990)* present the more advanced IHMS view for design which assigns complementary roles for the human and artificial components. They suggest the distribution of the intelligent function among human and artificial agents which can compete or cooperate at design time. A flexible IHMS design environment would enable input from the human designer inside the optimization loop at different levels of automation or implicit control to be decided at design time.

A shift towards the IHMS view is apparent from the growing number of practical applications in optimization which attempt to bring the two bounds closer or a setup that offers a given level of automation. *Hartman et al. (1995)* present a structural optimization system in which the manual loop can interrupt and override the automated loop, where they have tried to make switching between the two bounds relatively convenient. *Oates et al. (1994)* present a mixed initiative system for schedule maintenance in a simulated shipping network where they showed that the human and agent working together are able to achieve better results than either working alone. *Shahroudi (1994)* implemented a high speed human-in-the-loop design optimization of gas turbine engine concepts. On a larger scale a complex multidisciplinary design system would require a coordination of a number of semiautomatic agents which collaborate. *Olsen et al. (1994)* provide a formal treatment of the ontology whereby agents can communicate, which they plan to implement to multidisciplinary design optimization of satellites.

The proposed concept in this paper allows both cooperation and competition between the human designer and the numerical optimization agent. Semiautomatic control is provided by enabling the designer to modify design variables, simple bounds, constraint functions, the merit function and control parameters (or tuning parameters) which fix the numerical search strategy.

This translates into four modes of control:

- direct control on the search progress;
- direct control on the search problem;
- indirect control on the search problem;
- direct control of numerical search strategy.

To allow for these interactions, the design problem and the state of the design object are parameterized and coupled to graphical agents which allow visualization as well as communication of steering information from the human to the solver agent.

Implementation of these interactive optimization concepts in the Computational Steering Environment (*CSE*) takes the reader to the application domain where the theoretical expectations are realized in practice for three simple design examples.

2. THE CONCEPT

Regardless of the difficulty of defining merit functions and constraints, to qualify for a point in the semiautomatic spectrum we must have a mathematical expression of the design problem as shown below:

$$\begin{aligned} &\min_{X \in IR^n} f(X), X = x_1, x_2, \ldots, x_n \\ \text{subject to } & g_i(X) = 0 \text{ for } i = 1,2,\ldots,m_1 \\ & g_i(X) \geq 0 \text{ for } i = m_1 + 1, \ldots, m \\ \text{where } & f: IR^n \rightarrow IR \\ & g_i: IR^n \rightarrow IR \text{ for } i = 1,2,\ldots,m \end{aligned} \quad (1)$$

fully recognizing that this expression may be imperfect requiring improvement at design time.

The following subsections identifies four major processes inside a numerical optimizer. The intercommunication between the processes can in general be very complex, so a simple way to enable the steering of theses processes is to parameterize the problem statement and make these control parameters available for visual control. Several modes of control on the search progress, search problem and search strategy then become possible which offer an increasing degree of automation.

2.1 Interaction with Numerical Optimizer

The numerical optimizer agent can be visualized as consisting of four major processes which communicate with each other and with a data storage as shown in Fig. (2).

Depending on the complexity of the numerical agent, the inter process communications inside are in general complex and not necessarily as clean as illustrated, for example the "find a better state" process may call the other processes in different order and with different frequency as it finds necessary. Yet there appears to be four simple continuous interaction possibilities via the storage data that these processes utilize:

1. interaction with the current state *X*;
2. interaction with the constraints *g(X)*;
3. interaction with merit functions *f(X)*;
4. interaction with the numerical search strategy *S*.

These offer increasing levels of automation or increasing level of implicit control on design variables. For example suppose we are optimizing the weight (merit *f(X)*) of a turbine disk which is modeled in finite element from by say a million mesh points (design variables *X*) subject to a bound on allowable stress (constraint *g(X)*) using a genetic algorithm solver. As the optimizer proceeds the user can explicitly control the individual mesh points (low automation) to compete or collaborate with the numerical results. Controlling the allowable stress constraints can however affect a larger number of mesh points (increasing automation). Modifying the calculation of weight (e.g. by adjusting material density) can affect a much larger number of mesh points (still higher automation) because this affects the centrifugal forces that the disc width has to withstand. A higher level of control is still possible by controlling the *S* parameters that affect the diversification/intensification functions of the genetic search strategy.

It may appear at first sight that interaction with *S* is not comparable to others with regard to the automation level. However the numerical search strategy uses *X*, *g(X)* and *f(X)* to find the next best point. Therefore *S* offers a higher level of implicit control.

2.2 Search Problem Parameterized

The above continuous interactions are performed via controlling stored parameter values. The search problem can be parameterized in several ways but the most non-intrusive way is to allow the merit and constraint functions to look very close to their original form after some control parameters have been added as shown below:

$$\begin{aligned} &\min_{\substack{X \in IR^n \\ W \in IR^n}} f(X, W_f) \\ \text{where } & X = x_1, x_2, \ldots, x_n, W_f = w_{f_1}, w_{f_2}, \ldots, w_{f_N} \\ \text{subject to } & g_i(X, W_g) = 0 \text{ for } i = 1,2,\ldots,m_1 \\ & g_i(X, W_g) \geq 0 \text{ for } i = m_1 + 1, \ldots, m \\ \text{where } & f: IR^n \rightarrow IR \\ & g_i: IR^n \rightarrow IR \text{ for } i = 1,2,\ldots,m \\ & W_g = w_{g_1}, w_{g_2}, \ldots, w_{g_M} \end{aligned} \quad (2)$$

Where the numerical solver is only allowed to control or modify the *X* values. The added control parameters W_g and W_f are therefore parameters that have side effect but are not visible to the numerical solver. Consider the following example to illustrate the transformation,

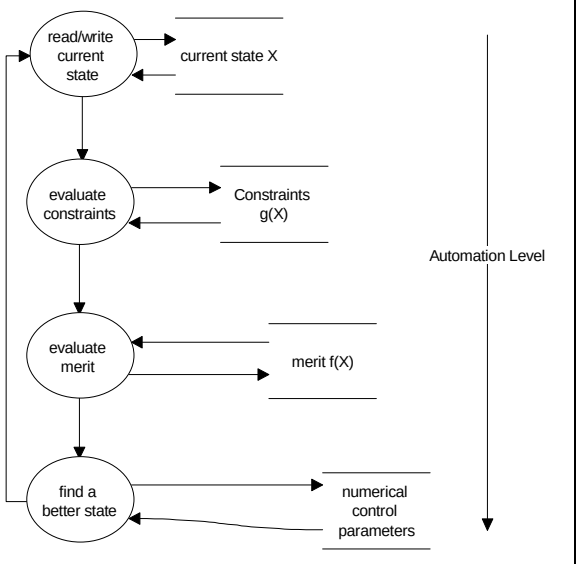

Figure 2: Processes in Optimization.

$$\min_{x \in IR^n} f(X) = 5x_3^{\,3} + 3x_2^{\,2} + 4x_1 + 5, \quad \text{subject to } g_1(X) = 3x_1^{\,2} + 5x_2^{\,2} + 1x_3^{\,2} = 0 \tag{3}$$

which can be re-written with parameters that have side effect,

$$\min_{x \in IR^n} f(X) = w_{f1}x_3^{\,3} + w_{f2}x_2^{\,2} + w_{f3}x_1 + w_{f4}, \quad \text{subject to } g_1(X) = w_{g1}x_1^{\,2} + w_{g2}x_2^{\,2} + w_{g3}x_3^{\,w_{g4}} = 0 \tag{4}$$

such that they enable the control of numerical solver which use these functions to operate.

2.3 Interaction via Graphic Agents

A good way to visualize and interact with the optimization is via graphic agents. Now that the search problem has been parameterized, it is possible to define *problem*, *optimum* and *numeric agents* which correspond merit+constraints, state of the optimum and numerical control parameters respectively as shown in Fig. (3). The responsibilities of these agents are:

- graphical representation of data to the user;
- receiving steering interaction from user to modify this data;
- broadcasting any modifications to the solver agent.

It is relatively easy to imagine how to define graphical representation of objects if it represents the shape of a physical object (e.g. the shape of an aircraft). However the more abstract the object, the more freedom exists to define a suitable graphical representation and the more difficult the task, which may cause disagreement between various practitioners. In the current implementation there is a possibility of adding user defined visualizations to standard ones in order to define the graphical aspect of the above agents.

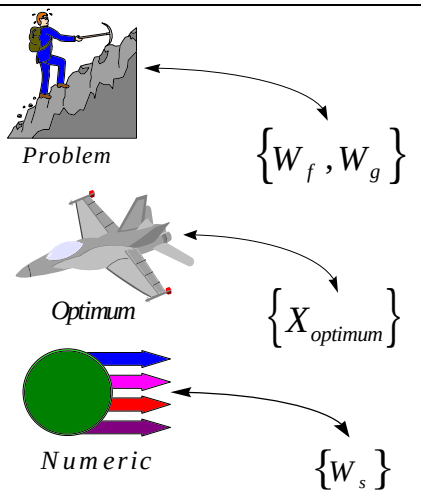

Figure 3: The Problem, Optimum and Numeric Agents.

Now that the search problem has been specified in parametric form and a decision is made on how to visualize these parameters and interact with them, we can proceed to set up the various interactive modes in semiautomatic optimization. The following sections describe how the above agents can be used in conjunction with a numerical optimizer to allow continuous in-the-loop user interaction.

2.4 Direct Control on the Search Progress

Fig. (4) shows the flow of *control* information for the overall semiautomatic setup. Arrows with a rounded base distinguish the flow of control information that is initiated by the human designer from others. The right and left sides of the figure show *Progress Control* and *Problem Control* respectively. The numerical solver is the central agent which requires four types of information to operate, namely *X*, *g(X)*, *f(X)* and *S*. This information comes from the *problem*, *optimum* and *numeric* agents which are under user control. The job of the solver is to continuously communicate a more optimum state for the design object X_{auto}.

As the numerical optimization progresses, new values of X_{auto} are continuously communicated to the mixer. If the user's hands are off the controls, the updates are flushed through to the *optimum* agent. In this way the user can monitor the progress by simply looking at the *optimum* agent. At any moment, the user may decide to interfere and provide new values of X_{user} by modifying the shape of the *optimum* agent. The mixer then mixes X_{user} and X_{auto} according to a mixture *ratio* which is also under human control. In this way the user can collaborate or compete with the numerical results depending on whether the requested change by the user is along or against the progress of the automatic results.

The advantages of direct control on the search progress are briefly:

- user intuition and experience is used at run time;
- location and direction of search can be continuously guided by the user with relative ease, as numerical optimization progresses;
- user can accelerate progress by remembering or imagining shortcuts;
- user can improve convergence by interfering, if the optimum is obviously going the wrong direction;
- freedom of user to influence location and direction of search, tends to globalize the optimization, although the numerical algorithm may be a local technique.

2.5 Direct and Indirect Control on the Search Problem

Given that the maximum complexity that the user can control is limited, there is a need for higher level

automation as the number of variables and amount of design detail increases. *Miller (1956)* quantifies the maximum number of independent effects that the human can keep track of to be around 7+/-2. Since the complexity of engineering applications is typically higher, we need a technique for controlling many variables by a few. Numerical optimization can be viewed as a method to control a lot of detail by specifying relatively compact criteria, or a technique for mapping a small problem to a large one. This view is not yet common among practitioners as there is often little or no support for changing these criteria at design time.

The left side of *Fig. (4)* shows in-the-loop control of the search problem. Here the user can interact by directly modifying the shape of the *problem* agent and hence the value of the control parameters that define the merit and constraint functions, as the numerical optimization proceeds. In other words, the user can modify or steer a relatively large part of the design object by modifying the criteria that results in the current state of the optimum via the solver. This represents a higher level of automation than the previous section because a larger number of design variables are controlled implicitly (by specifying a few things), instead of explicitly (by specifying a lot of things) .

The ability to modify the search problem and immediately observe the results in the progress of the optimum provides a flexible means to handle the always existent difficulty of defining a good set of constraints and merit functions for design. This is of crucial significance to the *Conceptual* and *Preliminary* phases of design for the following reasons:

- The human designer's knowledge about the design problem is relatively low which requires a lot of flexibility for modifying the design criteria until a point is reached when these criteria can become relatively fixed;
- The level of design detail is such that current computational and graphical resources are capable of computing and displaying the consequences of human interaction quickly. This allows the design activity to approach the *Natural Design Cycle* ideal (*Shahroudi (1994)*) which takes advantage of the superior short term capabilities of the human visualization pipeline;
- Nature of optimization is typically multidisciplinary or multiple objective in early design phases but the exact tradeoff between the various disciplines is not necessarily fixed, requiring flexibility to modify the relative importance of each discipline which fits in nicely with the concepts of this section.

The above design loop is in fact a *forward* design loop, i.e. starting with a criteria and ending with a better state of the design object in the loop. If this loop is high speed and the problem is simple, the user can reconstruct a mental reverse path and easily imagine what kind of specification is needed that results in a particular change in the design object. This is *inverse* design via mental reconstruction as discussed by *Shahroudi (1994)*.

Gruber (1991) presents a very interesting discussion of "acquisition of *design rationale*" where he attempts to reconstruct or capture the knowledge or reasoning underlying a design which is also a kind of inverse design. Example inverse design type questions are:

- What were the design decisions behind the Boeing 737 ?
- How does the Boeing 737 differ from Airbus A320 in terms of design criteria ? or What was the tradeoff between performance, cost and reliability that lead to the two aircraft designs ? How does this change if fuel prices increase ?

If the number of variables used in the definition of the merit function is large (e.g. because of too many disciplinary components) it may again become impractical to control them directly. To solve this problem and to enable inverse design type exercises, a still higher level of control is required where the merit control parameters are themselves steered implicitly via criteria.

For inverse design a general parametric inverter is required which translates the new state of the design object X_{user} into merit control parameters W_f as shown in *Fig. (4)*. The criteria that drives the inverter is to find the values of W_f which minimizes the difference between the actual and requested states of the design object X and X_{user} respectively, which is a constrained optimization problem in itself. The figure shows that inverse design is not yet implemented. Flow of control information for this indirect mode has also been largely omitted as it is out of the scope of this report. However there is a general purpose inverter which has been developed by *Shahroudi (1996)* for efficient multidimensional inverse computations of this

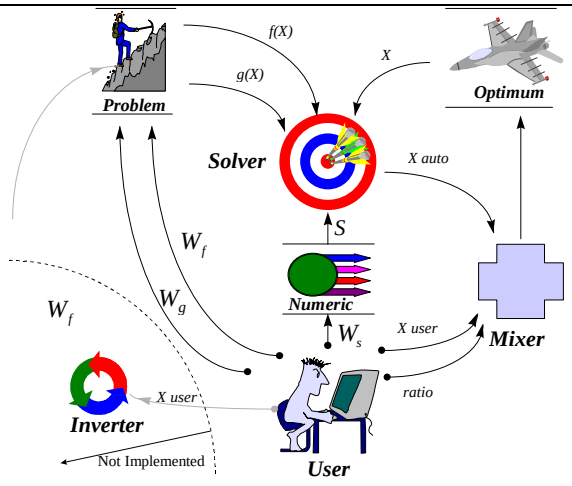

Figure 4: Flow of Control Information in the Semiautomatic Setup.

type, i.e. the tools required for this future extension are already available.

The potential benefits of in-the-loop direct and indirect user control on the search problem are briefly:

- alleviates some pressure on the precise statement of the design optimization problem, since these criteria can be varied and their consequences observed at design time;
- allows higher level of automation in design compared to that of previous section;
- particularly suitable for multidisciplinary or multiple objective design optimization, where it is interesting to adjust the weight of each discipline at run time;
- helps reconstruction of the design rationale underlying a particular design or for comparison between various designs.

2.6 Control on the Search Strategy

Interaction with the search strategy is a very interesting but difficult topic. There exist many numerical optimization algorithms which can be classified with attributes such as local, global , deterministic, probabilistic, greedy and so on. Each of these represents a different search strategy, precise details of which are usually known only to experts in each technique. The big question here is whether an average user (i.e. not a numerical optimization expert) is in general capable of independently deriving successful numerical search strategies for the problem at hand on the fly. In current systems, the user sets a search strategy by selecting a particular algorithm. Some guidance is available for making a good selection in the form of literature which compare various strategies for different problems e.g. *Baluja (1995)* or *Vanderplaats (1984)*.

```
#include "ad_model.h"
#include "../opt3/define_optim.h"
#include "../opt3/op_run.h"
void main()
{
   extern void evaluate_model(), init_model();
   …
/* initialize model  */
   init_model();
/* define optimization problem */
   OP_bgn_prob_def();
     OP_add_model("Aircraft", evaluate_model);
     …
/* indeps */
     OP_add_indep(&M, "M", &M_min, &M_max);
     …
/* non-linear constraints */
     OP_add_con(&b, "b", &b_min, &b_max);
     …
/* merit */
      OP_bgn_merit_def();
        OP_merit_minimize("mu_p", &mu_p);
        …
      OP_end_merit_def();
   OP_end_prob_def();
/* run semiautomatic optimization */
   OP_run();
}
```

Figure 5: Sample optimizer code for problem definition (aircraft design example).

High level run time strategy control is still an open question so the current implementation only allows low level control via the parameters provided by the numerical algorithm itself. In this way the search can be re-tuned by the designer as and when necessary. Tuning of algorithms at run time is a very important issue and there are *some self tuning* schemes which use feed back from the search progress, e.g. the *Reactive Tabu* by *Battiti et al. (1992)*. Tunable algorithms provide a gate for low level strategy control where it is possible to include the human in the tuning loop.

A tangible way to set up a higher level control is by specifying a super strategy which consists of a selection of algorithms which run in parallel and output their numerical result. The values of the strategy control parameters W_s then must consist of weight parameters which determine relative weight of each algorithm as well as their low level control parameters. However implementation of such a system requires a proper coordination as computational and memory requirements of various algorithms can vary a great deal which is left as a future issue.

A more rigorous approach to strategy control would require a language whereby the user can define custom numerical search strategies for the problem at hand at run time. *Tabu* search by *Glover et al. (1992)* is a move in this direction and is a simple meta specification for the search and consists of maintaining a *tabu* list (for places in the search space not to visit) derived from search history and a set of elementary moves which navigate the *neighborhoods* of a point. However the language is not rich enough to allow a user to specify the heuristics of how to maintain a tabu list and how to avoid places in this list and so on.

3. IMPLEMENTATION

This section explains the current implementation of the semiautomatic design optimization concept. All the modes of control described in previous sections have been implemented except indirect control on the search problem which awaits future implementation as explained before.

A simple library of functions were developed which allows easy definition of a design optimization problem in terms of any simulation (see section 3.1). Once the design problem is defined, a standard graphic user interface is automatically generated which includes the *Problem*, *Optimum* and *Numeric* agents. These can be used in

collaboration with application specific user interfaces to provide the various modes of control (see section 3.2).

The numerical routine used was the *E04UCF* from *NAG FORTRAN Library (1993)* which is a commercial software (see section 3.3).

The Computational Steering Environment (*CSE*) *(Wijk et al. (1994), Mulder et al. (1995)*) was used to enable collaboration of the numerical algorithm, graphical interface, simulation and human designer in a distributed fashion, like satellites around a central data manager (see section 3.4).

3.1 Specification of the Optimization Problem

To start an optimizer satellite, one must have a simulation which contains the relationship between variables of interest. Then the optimization problem has to be defined in terms of this simulation. A small library of functions were developed which allow this definition to be done in a few minutes. *Fig. (5)* shows a sample main program which was used to define the aircraft design problem discussed later in this report. Repetitive calls and some detail have been replaced with "…" . The variables that are used in the simulation *evaluate_model()* are declared in the header file *ad_model.h*. After telling the optimizer where to find the simulation *OP_add_model()*, the procedure selects which of the simulation variables are independent variables *OP_add_indep()*, constraints *OP_add_con()* or variables to be minimized *OP_merit_minimize()* or maximized *OP_merit_maximize()*. This gives the optimizer enough information to understand the design problem. There are also functions that defines parameters with side effects and their event handling. *OP_run()* starts the optimizer as a satellite around *DM* as shown later in *Fig. (8)*.

3.2 The User Implied and User Defined Interfaces

Defining all the necessary graphical objects from scratch is rather cumbersome. However the optimizer code contains enough information to automatically generate a standard user interface which contains the *Optimum*, *Problem* and *Numeric* agents. The user can modify this standard interface or supplement it with an application specific interface.

The automatically generated interface that was generated

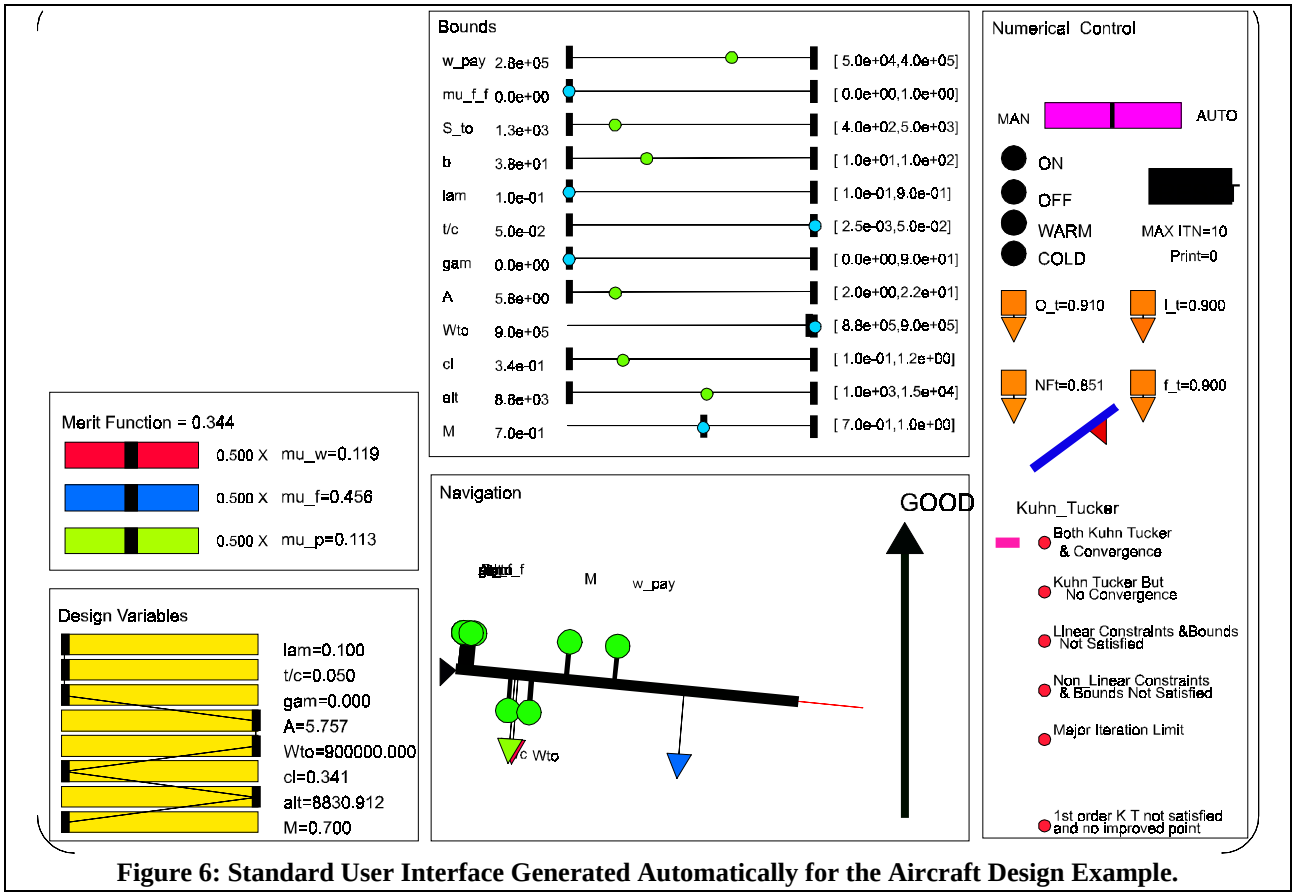

Figure 6: Standard User Interface Generated Automatically for the Aircraft Design Example.

for the sample code of *Fig. (5)* is shown in *Fig. (6)*.

The right hand side of this figure shows the *Numeric* agent. The lower half indicates the error condition of the solver. The flag is up if the Kuhn Tucker condition is satisfied. The middle part is coupled to four important tolerances used in the solver such as optimality and function tolerance. The top slider determines the mixing ratio for mixing manual and automatic results as discussed before. Warm and Cold switches are select two levels of accuracy and computational speed for the solver. As the optimization runs, the algorithm may get into an error condition. The user can then continuously tweak the tolerance levers until the Kuhn Tucker is satisfied. If the algorithm become too jumpy or unstable, the user can switch the solver to the more accurate Cold mode, increase the number of iterations or shift the top slider towards manual.

The *Bounds* box is part of the *Problem* agent which controls the constraint variables via sliders. The value of constraint variables is coupled to the position of the filled circles whose colors change from green to blue to red depending on whether a constraint is satisfied, active or unsatisfied respectively. The lower and upper bounds of a constraint are coupled to the position of filled rectangles at either end of each slider. Equal upper and lower bounds can be realized by shifting their corresponding rectangles to the same position, which is in fact an equality constraint. It should be easy to see that the user can easily drive the optimization problem by moving these bounds and observing the results, e.g. making active constraints inactive and changing the hardness of constraints etc.

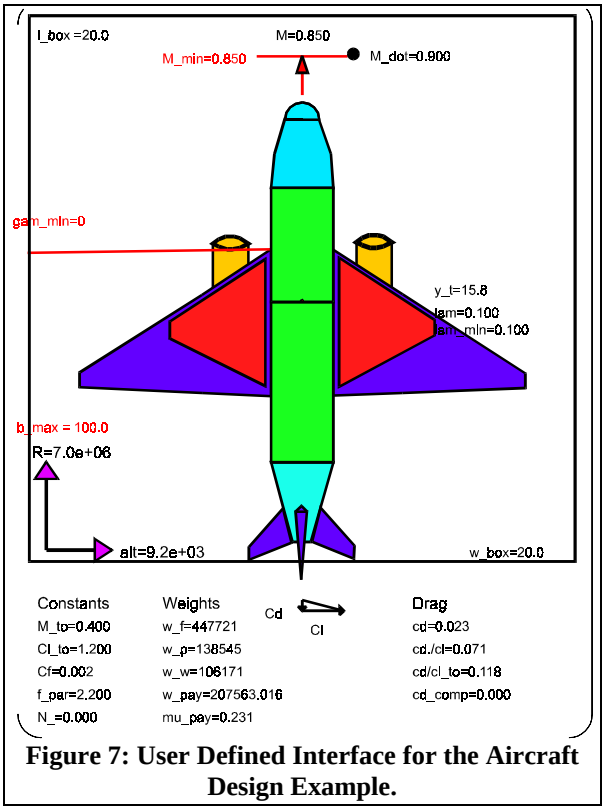

Figure 7: User Defined Interface for the Aircraft Design Example.

The *Merit* box on the left is also a part of the *Problem* agent. For the aircraft design example, three minimization criteria were specified in the optimizer code. Here a slider is assigned to the weight of each criterion such that shifting them sideways affects the trade-off between them. For example one can increase the weight or importance of fuel fraction (the middle blue slider) and observe how this effects the optimum.

A problem may require a number of merit elements and a large number of constraints. This can make it difficult to keep track of all the dynamic graphics. The *Navigation* box attempts to condense all this information onto a single body which is a hinged lever. The angle of the lever is mapped to the overall value of the merit and indicates which direction the search is going. The arrows are the equivalent of the sliders in the *Merit* box. Each balloon represents a constraint whose distance from the fulcrum is mapped to the sensitivity of merit function to the constraint value along the current search direction. The *Navigation* box allows the user to modify the merit function and observe how the relative importance of various constraints and weights develops.

The *Design Variables* box is a group of sliders which form the *Optimum* agent. As the optimization progresses the position of these sliders change. The user can try to collaborate or compete with this progress depending on whether he shifts the slides along or against the direction of automated results. While this allows user interaction, it is not in general easy to make physical sense of how the optimum state is progressing. For this reason the user can add more application specific visualizations.

Fig. (7) shows the user defined interface for the aircraft design example. Here the *optimum* agent is the shape of the aircraft and it is much easier to monitor and interact with the progress of optimization. The user defined interface is also very useful for including constraints (e.g. minimum wing sweep angle) and all the parameters of interest which have a side effect but which are missing from the specification of the design problem. For example the range of the aircraft was not included in the optimizer code but several constraints and merit function depend on its value.

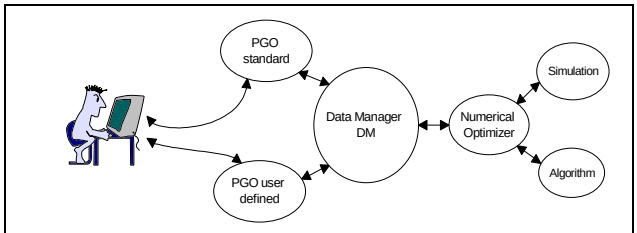

Figure 8: Data flow in the semiautomatic setup in *CSE* at run time.

The example interface allows the user to modify the range by pulling on the arrow (lower left of figure), which in turn results in a new optimum shape of aircraft.

3.3 Choice of Numerical Algorithm

The numerical algorithm *E04UCF* from *NAG FORTRAN Library (1993)* was selected which is a local search strategy based on sequential quadratic programming. It is a very reliable algorithm that can handle nonlinear constraints without user supplied gradient information. The setup would have been much more powerful had we chosen an optimization package such as *DOC/DOT* by *Vanderplaats (1995)* which offers a range of algorithms with a single problem definition interface. Time constraints however forced the selection of a single routine for the first proof of concept implementation.

3.4 The Semiautomatic Optimization Setup in *CSE*

The structure of the Computational Steering Environment (*CSE*) allows several programs (called satellites) running on different computers to collaborate via a high speed link to a central Data Manager (*DM*). One important satellite is the PGO (Parameterized Graphic Objects) editor which in edit mode allows the user to define graphic objects. To each graphic object, degrees of freedom can be attached which can be bound to variables in the data manager. In run mode, the shape of these objects reflect the values of variables in the *DM* and can be used for visualization as well as steering of simulations which communicate with the *DM* (see *Wijk et al. (1994)* for details).

Fig.8 shows how the semiautomatic concept was implemented in *CSE*. A simulation provides the relationship between all variables of interest. The numerical optimizer satellite uses this together with the design problem and the current state of the design object to find a better state. This is updated to the *DM*, which in turn flushes the new data to the standard and user defined interfaces. The interfaces which contain the *problem*, *optimum*, *numeric* and user defined agents, continuously display the new situation. The user exercises control by modifying the shape of these agents, which is flushed through to the numerical optimizer via the *DM*.

In a purely distributed concept, the simulation and optimizer would be separate satellites which communicate via the *DM*. In practice however the numerical optimizer will call the simulation so frequently that a function call is much more efficient.

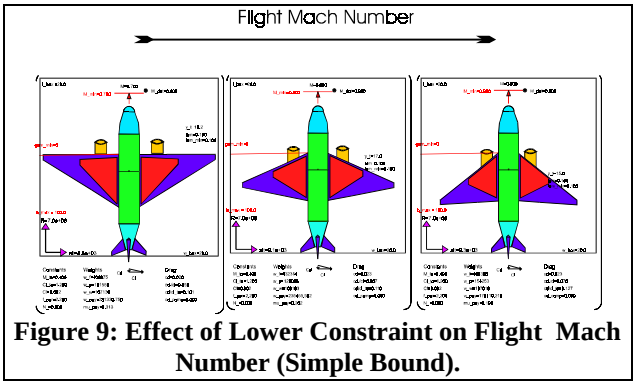

Figure 9: Effect of Lower Constraint on Flight Mach Number (Simple Bound).

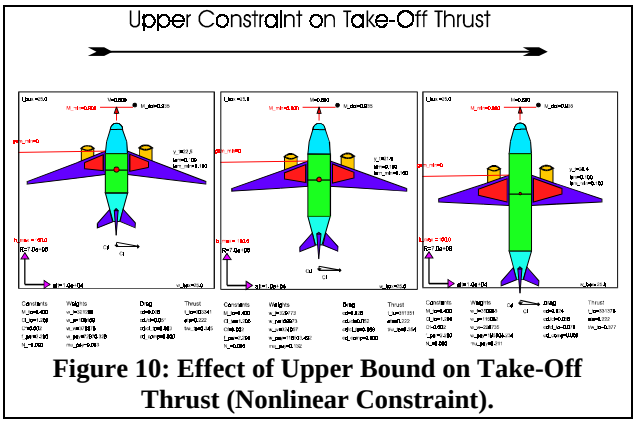

Figure 10: Effect of Upper Bound on Take-Off Thrust (Nonlinear Constraint).

4. APPLICATIONS

This section presents three examples for testing the semiautomatic optimization approach of this paper. The nature of these implementations is highly dynamic and interactive. The complexity of the examples are such that the user receives an information update many times a second and can continuously interact with graphical objects. This means that it is possible to generate a huge set of pictures, depending on the exact location in the search space, the sequence of user interactions, values of

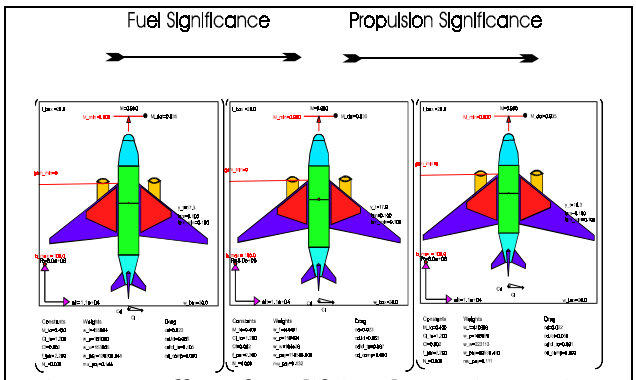

Figure 11: Effect of Modifying the Merit Function.

the constraints, various trade-off positions in the merit function and so on.

The main purpose for including these examples is to show the flexibility of the semiautomatic approach. The discussion is therefore necessarily concise, citing only a few example interactions and omitting detailed explanation of the optimization results.

4.1 Aircraft Conceptual Design

This is a multidisciplinary conceptual design exercise which is in essence similar to that presented by *Johnson (1988)*. A brief specification of the design problem is:

Independent or Design Variables: Flight Mach Number, Altitude, Lift Coefficient, Take-Off Weight, Aspect Ratio, Wing Sweep Angle, Wing Thickness to Chord Ratio and Wing Taper Ratio.

Simple Bounds: on all the design variables above.

Constraints: Wing Span, Take-Off Filed Length, Fraction of Fuel Stored in Wings, Payload Weight and Take-Off Thrust.

Composite Merit Function: minimize Fuel Weight Fraction, Propulsion Weight Fraction and Wing Weight Fraction with respect to Take-Off Weight.

Parameters with Side Effect: long list including Range, Technology Factors, Parasitic Drag Area, Take-Off Lift Coefficient etc.

For derivation of the three separate elements of the merit function above see *Torenbeek (1992).*

With the optimization in run mode, *Fig.9*. shows that increasing the flight Mach number results in a more sporty look for the optimum aircraft, i.e. the wings sweep back , bigger engines etc. It is important to note that this transition only takes fractions of a second and the user can study the effect of the other parameters (e.g. Range) by simply modifying the corresponding picture.

In *Fig. 10*, the user interacts at a higher automation level in order to learn the effect of changing the upper bound on the value of take-off thrust which indicates the size of the engines.

Fig. 8 and *Fig 9* show the state of the optimum if the three minimization criteria that form the merit function are equally significant. But the correct trade-off depends on many factors such as fuel and engine prices. *Fig. 11* shows user interaction at a still higher level of automation to answer questions related to the trade-off.

4.2 Mars Rover Route Optimization

In order to explore the Mars environment, a rover vehicle has to follow a trajectory on the surface, which ends in a target area while avoiding obstacles along the way. The Rover may generate power via solar cells along the trajectory or use the stored energy in the batteries. In either case, it is important to optimize the trajectory for total energy expenditure, total distance traveled or a combination.

For this purpose a simple simulation was developed by the author which models a 3D hilly Martian terrain with obstacles together with a rover which follows a multiple segment trajectory. Each segment of the trajectory is a straight line when viewed from above, but follows the terrain surface. The simulation calculates the driving and breaking power as well as the distance traveled for a constant speed rover. A brief specification of the optimization problem is:

Independent or Design Variables: the points that define the trajectory except the starting point.

Simple Bounds: the trajectory must remain inside a rectangular area. The end point of the trajectory must be inside the target area.

Constraints: the trajectory must avoid all obstacles.

Composite Merit Function: minimize magnitude of total Driving Energy, magnitude of total Breaking Energy and total trajectory Distance.

Parameters with Side Effect: Number of segments in a trajectory, Friction Coefficient of Martian surface, Martian Terrain Model.

Fig. 12 shows the user defined interface for this problem. The altitude information has been mapped to a color ramp so that blue, green and red indicate low, medium and high altitude respectively. The terrain is defined by a number of hill. Here high and low *P* values represent sharp and flat hills respectively. The user can

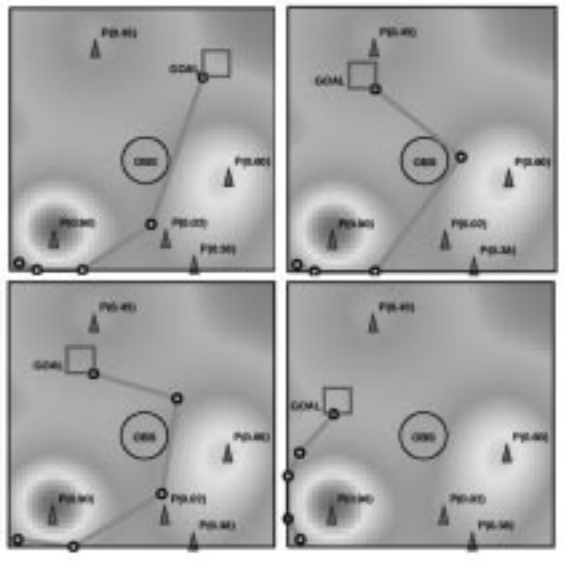

Figure 12: Optimum Trajectories as Target Moves.

modify the terrain, the target area, the trajectory, the size and position of obstacles. The figure shows that as the user moves the target area, the solver updates the trajectory according to the optimization criteria specified above.

Fig. 12 also serves as a good illustration of how easy it is to fool a high quality numerical routine. There are of course many solutions possible and the solver tends to find the best one that is the closest and not a global optimum, since it uses a local search strategy. The user on the other hand does not have this limitation and at some point can easily guess that a better solution must be possible going clockwise around the high peak. The lower right solution of the figure was obtained by the user dragging the trajectory towards the left side of the high peak, i.e. the user competed with the numerical results. Once the trajectory was roughly on the correct side of the peak, the solver and the user collaborated to polish the trajectory to the state shown in the figure. The semiautomatic approach therefore tends to globalize the local search technique because the user can drive the search into different locations of the search space in a collaborative or competitive capacity.

Depending on the particular mission, the tradeoff between the traveled distance and expended energy may differ. For example it may be important to reach the target area more quickly at some expense to energy. *Fig. 13* shows the results of a tradeoff study that the user can easily perform by in-the-loop interaction.

Naturally, the optimum trajectory depends on the surface terrain which the user can modify by dragging the peaks or adjusting their value. *Fig. 14* shows the results of this exercise.

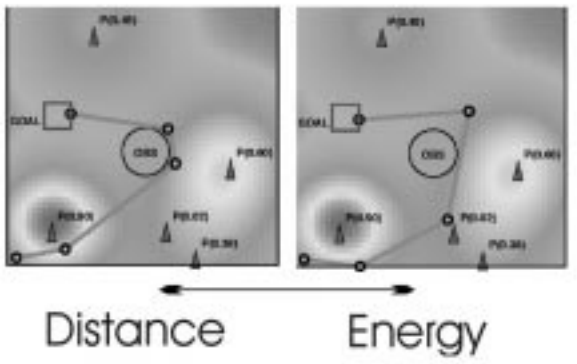

Figure 13: Tradeoffs in Merit Function.

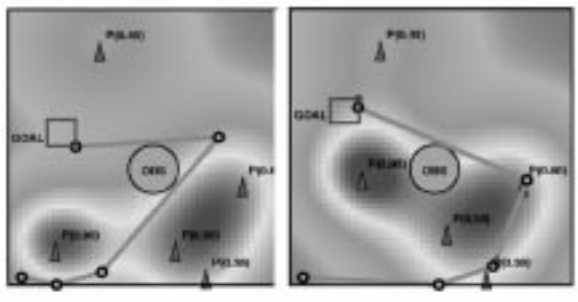

Figure 14: Effect of Terrain.

4.3 Multistage Rocket Configuration

Cornelis et al. (1979) divide a multistage rocket into several sub-rockets such that a sub-rocket is considered as payload of the surrounding sub-rocket and so on. Using these relations, one can specify a simple design problem for multistage rockets as follows:

Independent or Design Variables: mass ratios of each sub-rocket.

Simple Bounds: mass ratios must remain within known practical limits.

Constraint: the rocket reach a minimum required energy altitude.

Composite Merit Function: maximize overall

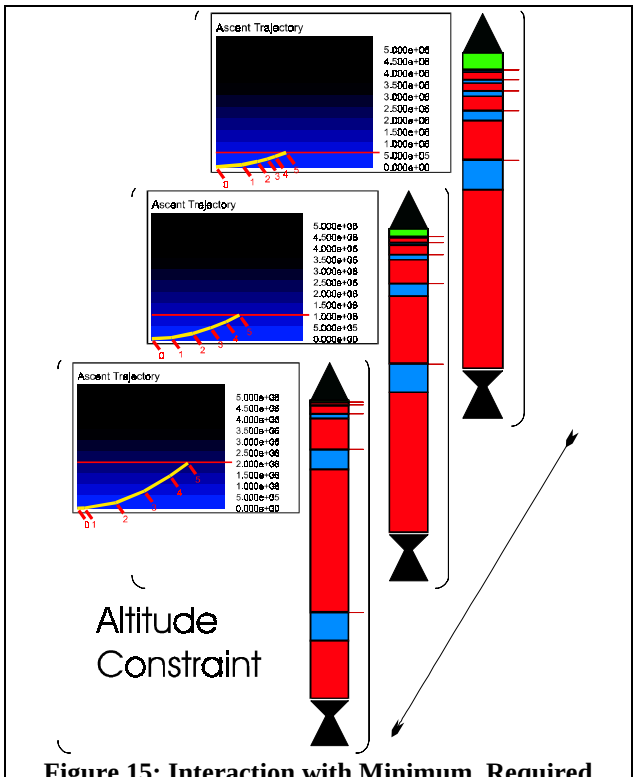

Figure 15: Interaction with Minimum Required Energy Height.

payload weight while minimizing overall structural and fuel weight.

Parameters with Side Effect: number of stages, fuel type and exhaust velocity of each stage.

Fig. 15 shows the user defined interface for this problem. Red, blue and green on the rocket represent propellant, structure and payload weight respectively. The ascent trajectory to the left of the rocket shows the energy height reached after each stage is jettisoned. The vertical position of the horizontal red bar is coupled to the minimum required energy height. The user can continuously steer this constraint and observe the consequences in the configuration of the rocket as shown.

When altitude constraint is not so stringent, there are many local optima possible requiring the user to drag the rocket away from a given optimum to find new ones. *Fig. 16* shows three such optima which were found by the user dragging the design variables towards a simpler configuration (i.e. smaller number of stages) while the solver was running.

5. CONCLUSIONS

There has been a growing shift towards the Integrated Human Machine Systems view of design optimization, both in concept and practice. The optimum position in the automation spectrum depends on many factors such as the quality of the designer, the complexity of the problem, the current focus and the computational characteristics. All these factors typically vary as design optimization progresses, therefore an ideal system should allow user interaction in the complete automation spectrum including the fully manual and fully automated bounds.

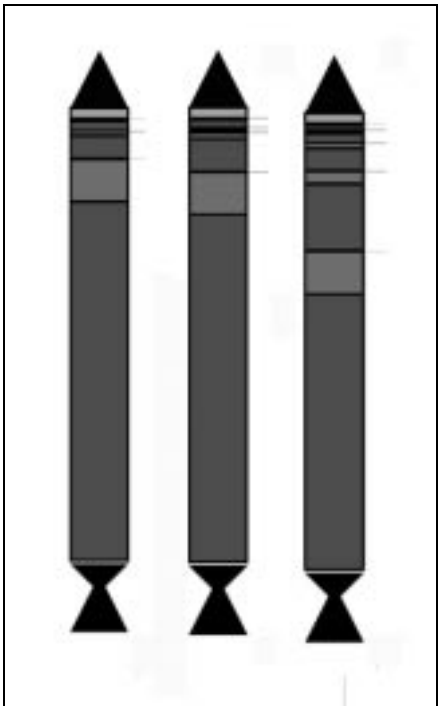

Figure 16: User Drives the Solver to Find Alternative Optima.

This paper concentrates on the semi-automatic gray area of this spectrum and presents a setup where a user interacts with the design optimization loop by steering the *Problem*, *Numeric* and *Optimum* agents in order to control the *search progress* and the *search problem*.

Including the human in the optimization loop, allows us to draw from the best of the two worlds of manual and automatic optimization. The benefits are briefly:

- It alleviates some pressure on the precise statement of the design optimization problem, since these criteria can be varied and their consequences observed at design time;
- It is particularly suitable for multidisciplinary or multiple objective design optimization, where it is interesting to study the tradeoff between various discipline at run time;
- It aids the reconstruction of the design rationale underlying a particular design or for comparison between various designs;
- User intuition and experience is used at run time. The location and direction of search can be continuously guided by the user in collaboration or competition with the numerical results;
- Freedom of the user to influence the search progress at run time, tends to globalize the optimization, although the numerical algorithm may be a local one.

To take advantage of the current implementation the user starts with a simulation. The optimization problem is then specified in terms of this simulation by using a purpose made library of functions, which typically takes a few minutes to complete. Thereafter, a standard graphic user interface is automatically generated which includes the *Problem*, *Optimum* and *Numeric* agents. These agents are responsible for graphic representation as well as communication of steering information from the user to the solver. The standard interface is then used in collaboration with a user defined interface to provide the various modes of control discussed above.

Finally, the potential of the semi-automatic approach has been demonstrated by means of three human-in-the-loop optimization examples.

ACKNOWLEDGEMENTS

This work is supported by the Netherlands Computer Science Research Foundation (SION), with financial support of the Netherlands Organization for Scientific Research (NWO).

REFERENCES

Baluga, S. (1995). *An empirical Comparison of Seven Iterative and Evolutionary Function Optimization Heuristics*, CMU-CS-95-193, CMU, Pittsburgh, PA.

Battiti, R. et al. (1994). *The Reactive Tabu Search*, ORSA Journal on Computing Vol. 6, N.2.

Boy G. et al. (1990). *Intelligent Assistant Systems: Support for Integrated Human-Machine Systems*, Proceedings of 1990 AAAI Spring Symposium on Knowledge Based Human-Computer Communication, Stanford University.

Cornelis, J. W. et al. (1979). *Rocket Propulsion and Spaceflight Dynamics*, Pitman, p. 262-281.

Glover F. et al. (1992), *Tabu Search*, Modern Heuristic Techniques for combinatorial problems, Blackwell Publishing.

Gruber, T. R. (1991), *Interactive Acquisition of Justifications: Learning "Why" by Being told "What"*, To appear in IEEE Expert, based on a lecture presented at the First Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop, Kyoto, Japan, 1990.

Hartman, D. et al. (1995*), Object-Oriented Structural Optimization Models*, OPTI 95 - 4th International Conference on Computer Aided Optimum Design of Structures, Miami, Florida, 19-21 September 1995.

Johnson, V. S. (1988), *Optimizing Conceptual Aircraft Designs for Minimum Life Cycle Cost*, Proceedings of the second NASA/Air Force Symposium on Recent Experiences in Multidisciplinary analysis and optimization, Hampton, VA.

Miller, G. (1956). *The Magical Number Seven, Plus or Minus Two. Some Limits on Our Capacity for Processing Information.* The Psychological Review Vol. 63(2).

Mulder, J. et al. (1995). *3D Computational Steering with Parameterized Graphic Objects.* Proceedings IEEE Visualization `95.

Numerical Algorithm Group (1993). *NAG FORTRAN Library User Manual Mark 16.*

Oates T. et al. (1994). *Human Plus Agents Maintain Schedules Better than Either Alone*, Computer Science Technical Report 94-03, University of Massachusetts, Amherst, MA.

Olsen, G. R. et al. (1994). *Collaborative Engineering based on Knowledge Sharing Agreements*, Proceedings of the 1994ASME Database Symposium, Minneapolis, MN.

Shahroudi, K.E. (1994). *Development and Validation of a Computer Assisted Design Methodology for Gas-Turbine-Based Aircraft Engines*. Delft University Press, ISBN 90-407-1070-8.

Shahroudi, K.E. (1996). *The Flipping Analytical Coin Tool: Closing The Information Flow Loop in High Speed Analysis*. Proceedings of the 2nd International Conference on Inverse Methods, Nantes, France.

Torenbeek, E. (1992). *Optimum Wing Area, Aspect ratio and Cruise Altitude for Long Range Transport Aircraft.* Report LR-775, Delft University of Technology.

Vanderplaats, G.N. (1995) *DOC/DOT User Manual*, Vanderplaats Research & Development Inc.

Vanderplaats, G.N. (1984). *Numerical Optimization Techniques for Engineering Design*, McGraw-Hill.

Wijk, J.J. van et al. (1994), *An Environment for Computational Steering*. Presented at the Dagstuhl Seminar on Scientific Visualization, 23-27 May 1994, Germany. Proceedings to be published.